

Privacy Design Strategies

Jaap-Henk Hoepman

October 25, 2012

Abstract In this paper we define the notion of a privacy design strategy. These strategies help to support privacy by design throughout the full software development life cycle, even before the design phase. Using current data protection legislation as point of departure we derive the following eight privacy design strategies: MINIMISE, HIDE, SEPARATE, AGGREGATE, INFORM, CONTROL, ENFORCE, and DEMONSTRATE. We show that these design strategies provide a useful classification of privacy design patterns and the underlying privacy enhancing technologies, by validating them against two different models of ICT systems, as well as existing privacy principles.

1 Introduction

Privacy by design [5] is the philosophy of protecting privacy throughout the process of technological development, that is from the conception of a new technology up to its realisation. The idea is that when privacy is a integral part of the technological development process, the final product protects privacy throughout its entire life cycle.

In the context of developing IT systems, this implies that privacy protection is a system requirement that must be treated like any other functional requirement. In particular, privacy protection (together with all other requirements) will determine the design and implementation of the system. To support privacy by design, we therefore

need guiding principles to support the inclusion of privacy requirements throughout the system development life cycle¹, in particular during the concept development, analysis, design and implementation phases. Unfortunately (cf. Gürses *et al.* [14]) there is so far little experience in applying privacy by design in engineering. This paper aims to contribute to closing this gap.

An important methodology during the design phase is the application of so called software design patterns to refine the system architecture to achieve certain functional requirements within a given set of constraints. In particular, some privacy design patterns have recently been proposed in the context of privacy protection. However, such design patterns do not necessarily play a role in the earlier, concept development and analysis, phases of the software development cycle. The main reason is that such design patterns are already quite detailed in nature, and more geared towards solving an implementation problem. To guide the development team in the earlier stages, privacy design strategies at a higher level of abstraction are needed.

Our approach extends the work by Spiekermann and Cranor [27] by providing system developers concrete strategies to actually engineer privacy. The strategies we propose cover both the privacy-by-policy and privacy-by-architecture approach from Spiekermann and Cranor [27]. Whereas they see these two approached as essentially mutually exclusive (a system that is engineered as privacy-by-architecture does not process privacy sensitive data and therefore does not need privacy-by-policy) our view is less binary: a system architecture will hardly ever guarantee full privacy, and a privacy policy alone does not give sufficient privacy guarantees either.

Jaap-Henk Hoepman
TNO
PO. Box 1416, 9701 BK Groningen, The Netherlands
jaap-henk.hoepman@tno.nl
and
Institute for Computing and Information Sciences (ICIS)
Radboud University Nijmegen
PO. Box 9010, 6500 GL Nijmegen, the Netherlands
jhh@cs.ru.nl.

¹ http://en.wikipedia.org/wiki/Systems_development_life-cycle

In this paper we define the notion of a privacy design strategy, and derive the following eight privacy design strategies: MINIMISE, HIDE, SEPARATE, AGGREGATE, INFORM, CONTROL ENFORCE and DEMONSTRATE based on both the legal and the technical perspective on privacy protection. We validate our approach by showing how these strategies apply to both an information storage and information flow type of system, and by comparing our classification to existing privacy frameworks. We believe these strategies help to support privacy by design throughout the full software development life cycle, even before the design phase, by making explicit which high level strategies can be applied to protect privacy when drafting the first concepts from which a new information system will be derived.

2 On design strategies, design patterns and privacy enhancing technologies

In privacy by design, privacy enhancing technologies and privacy design patterns play an important role, but their distinction, and their role during the system development life cycle, is not always clear. Therefore we need to make precise what the differences are between (privacy) design strategies, (privacy) design patterns, and privacy enhancing technologies. We do so in this section, from the software architecture perspective.

Software architecture encompasses the set of significant decisions about the organisation of a software system², including the

- selection of the structural elements and their interfaces by which a system is composed
- behaviour as specified in collaborations among those elements
- composition of these structural and behavioural elements into larger subsystem
- architectural style that guides this organisation

2.1 Design patterns

The concept of a *design pattern* is a useful vehicle for making such decisions. A design pattern

“provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly recurring structure of communicating components that solves a general design problem within a particular context.” [3]

² Based on an original definition by Mary Shaw, expanded in 1995 by Booch *et al.* [21].

Typically, the description [10] of a design pattern contains at least its name, purpose, description of the application context, its structure, implementation (components and their relationships), and the consequences (results, side effects and trade offs) of applying the pattern. Many design patterns exist, at varying levels of abstraction. A classical design pattern is the *Model-View-Controller* pattern³, that separates the representation of the data (the model) from the way it is represented towards the user (the view) as well as how the user can modify that data (controller). A much simpler design pattern is the *Iterator* pattern, that “provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation” [10]. By using this pattern, the actual implementation of the list to process has become irrelevant and can be changed without changing higher level code.

Few privacy design patterns have been explicitly described as such to date. We are aware of the work of Hafiz [15,16], Pearson [24,23] and a recent initiative of the UC Berkeley School of Information⁴. Many more privacy design patterns exist though, although they have never been described as such. Sweeney’s *k-anonymity* concept [28] is a classical example of an idea that implicitly defines a privacy design pattern. Also the concept of a *zero knowledge proof* [12] can be viewed as a design pattern. In fact many of the privacy enhancing technologies (described below) implicitly define a corresponding privacy design pattern. Good examples are patterns like *attribute based credentials* (as studied in for example the ABC4TRUST project⁵) and *mix networks*. This is discussed further below.

2.2 Design strategies

Because certain design patterns have a higher level of abstraction than others, some authors also distinguish *architecture patterns*, that

“express a fundamental structural organisation or schema for software systems. They provide a set of predefined subsystems, specify their responsibilities, and include rules and guidelines for organising the relationships between them.”⁶

³ Originally formulated in the late 1970s by Trygve Reenskaug at Xerox PARC, as part of the Smalltalk system.

⁴ <http://privacypatterns.org/>

⁵ www.abc3trust.eu

⁶ See <http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns.html>, and The Open Group Architecture Framework (TOGAF) <http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap28.html>

The *Model-View-Controller* pattern cited above is often considered such an architecture pattern. The distinction between an architecture pattern and a design pattern is not always easily made, however. Moreover, there are even more general principles that guide the system architecture without imposing a specific structural organisation or schema for the system.

We choose, therefore, to express such higher level abstractions in terms of *design strategies*. We define this as follows.

A design strategy describes a fundamental approach to achieve a certain design goal, that has certain properties that allow it to be distinguished from other (basic) approaches that achieve the same goal.

For example, the construction of a bridge may be classified depending on how the forces of tension, compression, bending, torsion and shear are distributed through its structure. A very strategic decision is to decide whether to use a form of *SUSPENSION* (where the deck is suspended from below a main cable) instead of using a more classical form of *SUPPORT* (for example, using arches). A *privacy design strategy* then is a design strategy that achieves (some level of) privacy protection as its goal.

Design strategies do not necessarily impose a specific structure on the system (although they certainly limit the possible structural realisations of it). Therefore, they are also applicable during the concept development and analysis phase of the development cycle⁷.

2.3 Privacy enhancing technologies

Privacy Enhancing Technologies (PETs) are better known, and much more studied. Borking and Blarckom *et al.* [1, 30] define them as follows.

“Privacy-Enhancing Technologies is a system of ICT measures protecting informational privacy by eliminating or minimising personal data thereby preventing unnecessary or unwanted processing of personal data, without the loss of the functionality of the information system.”

This definition was later adopted almost literally by the European Commission [8]. It is slightly biased towards the data-minimisation principle, and is also a bit more high-level than the types of privacy enhancing technologies⁸ typically studied.

⁷ We note that the notion of a privacy design strategy should not be confused with the foundational principles of Cavoukian [5] or the concept of a privacy principle from the ISO 29100 Privacy framework [19].

⁸ See for example the annual Privacy Enhancing Technologies Symposium <http://petsymposium.org/>.

In principle, PETs are used to implement a certain privacy design pattern with concrete technology. For example, both ‘Idemix’ [4] and ‘u-prove’ [2] are privacy enhancing technologies implementing the (implicit) design pattern *anonymous credentials*. There are many more examples of privacy enhancing technologies, like ‘cut-and-choose’ techniques [7], ‘onion routing’⁹ [6] to name but a few (and see also [11]).

2.4 Discussion

An earlier draft of this paper confused anonymous credentials for a privacy enhancing technology. This is wrong. In fact *attribute based credentials*¹⁰ are a perfect example of a privacy design pattern¹¹. This pattern describes a general structure separating users, issuers, and verifiers, where the link between issuing a credential and proving possession of it is broken to prevent tracking users. This is a telling testimony to the fact that the distinction between a privacy design pattern and a privacy enhancing technology is not so easy to make in practice (if only because historically many design patterns are only implicitly defined by the corresponding privacy enhancing technology).

Let us try to make the distinction clearer by reconsidering the example of the construction of a bridge introduced above. In terms of this example, a design strategy is the use of *SUPPORT* (instead of *SUSPENSION*). There are several design options after one has decided to go for the *SUPPORT* strategy. One of these options is the use of arches to create the structural support required. *Arches* then is a design pattern - in fact a design pattern that occurs in many different constructions beyond bridges. A concrete technology to build (i.e., implement) an arch bridge is to use bricks for the actual construction of a so-called ‘round arch’.

Design patterns may overlap, and may vary in the level of detail they provide. Similar to the difference in abstraction between the *Model-View-Controller* and the *Iterator* pattern, the *attribute based credentials* pattern is much more concrete than a more generic *use pseudonyms*

⁹ Made popular through the TOR project <http://www.torproject.org/>.

¹⁰ Attribute based credentials is a better term than anonymous credentials because in many cases the credential may contain non-anonymous information.

¹¹ In this paper we will occasionally refer to a design pattern as if it is already properly defined as such. This is often not the case (including the case of the *attribute based credential* at hand here). For now we will just appeal to the intuitive understanding of the main structure of the pattern, and defer a full description of such a pattern to further research. In a way, the secondary purpose of this paper is to identify such new privacy design patterns, and to merely record their existence for now.

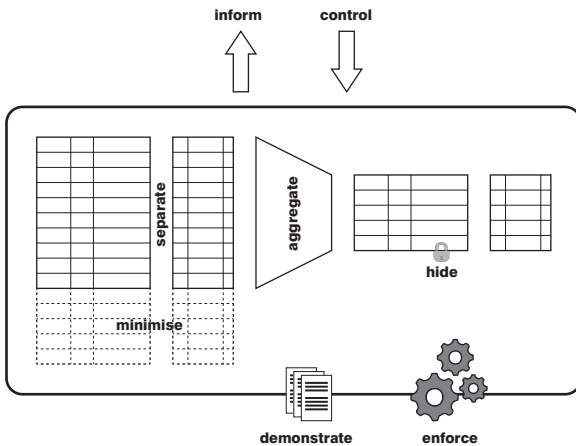


Fig. 1 The database metaphor of the eight privacy design strategies.

pattern, and in fact may partially overlap with that more generic pattern.

We also note that a privacy design pattern may sometimes implement several privacy design strategies. For example the *use pseudonyms* design pattern both implements the MINIMISE strategy and the SEPARATE strategy (as will become apparent when we have described the different strategies in detail later in section 4). Similarly, a privacy enhancing technology may be applicable within several different privacy design patterns.

3 Deriving privacy design strategies

A natural starting point to derive some privacy preserving strategies is to look at when and how privacy is violated, and then consider how these violations can be prevented. For example, Solove's taxonomy [26] identifies four basic groups of activities that affect privacy: information collection, information processing, information dissemination and invasions¹². He then discusses in detail the different ways in which certain specific activities (like surveillance, aggregation, disclosure, and intrusion) should be understood and dealt with from the legal perspective. The specific activities identified by Solove are too fine grained. Although they may in fact be interesting to distinguish from a legal perspective, many of them involve basically the same methods at a technical level. His general subdivision however inspired us to look at IT systems at a higher level of abstraction to determine where and how privacy violations could be prevented.

In doing so, we can view an IT system either as an information storage system (i.e., database system) or an information flow system. Many of today's systems (e.g.,

classical business or government administration systems, but also social networks) are database systems. But if the sheer volume of data (for example those created by sensors and data in the Internet of Things) becomes too large to store, information flow systems that no longer store data but process it for immediate use offer an alternative view. Interestingly, as we shall see later, both views on IT systems are subject to the same eight privacy design strategies.

Let us first consider the information storage system view, also because current data protection legislation [9] is pretty much written with that model of an IT system in mind. In a database, information about people is stored in one or more tables. Each table (each with its own access conditions) stores certain sets of attributes about the people in the database. Sometimes, data is not stored at the level of individual persons, but is instead aggregated based on certain relevant group properties (like postal code). Within the legal framework, the collection of personal information should be proportional to the purpose for which it is collected, and this purpose should not be achievable through other, less invasive means.

In practice, this means that data collection should be *minimised*, for example by not storing individual rows in a database table for each and every individual, and the number of attributes stored should correspond to the purpose. Data collected for one purpose should be stored *separately* from data stored for another purpose, and linking of these database tables should not be easy. When data about individuals is not necessary for the purpose, only *aggregate* data should be stored. Personal data should be properly protected, and strict access control procedures should *limit access* to authorised persons only. A data subject should be *informed* about the fact that data about her is being processed, and she should be able to request modifications and corrections where appropriate. In fact the underlying principle of information self-determination dictates the she should be in *control*. Finally, the collection and processing of personal data should be done in accordance to a privacy policy, that should be actively *enforced*. The current proposal for the revision of the European privacy directive (into a regulation) also stresses the fact that data controllers should be able to *demonstrate compliance* with data protection legislation.

Given this analysis from the legal point of view, we see we can distinguish the following eight privacy design strategies: MINIMISE, SEPARATE, AGGREGATE, HIDE, INFORM, CONTROL, ENFORCE and DEMONSTRATE. A graphical representation of these strategies, when applied to a database system, is given in Figure 1.

¹² This is similar to the distinction made between data transfer, storage and processing by Spiekermann and Cranor [27].

4 The eight privacy design strategies

We will now proceed to describe these eight strategies in a bit more detail. After that we will show that this subdivision also makes sense from other perspectives.

4.1 Strategy #1: MINIMISE

The most basic privacy design strategy is data minimisation, which states that

The amount of personal information¹³ that is processed should be minimal.

This strategy is extensively discussed by Gürses *et al.* [14]. By ensuring that no, or no unnecessary, data is collected, the possible privacy impact of a system is limited. Data minimisation can take two forms: either a yes/no decision to collect any information about certain individuals is made (as a consequence, for some people no information will be collected at all), or the amount of information that is collected about each person is restricted to a limited set of characteristics.

Common design patterns that implements this strategy are *select before you collect* [20], *anonymisation* and *use pseudonyms* [25].

4.2 Strategy #2: HIDE

The second design strategy, HIDE, states that

Any personal information that is processed should be hidden from plain view.

The rationale behind this strategy is that by hiding personal information from plain view, it cannot easily be abused. The strategy does not directly say from whom the data should be hidden. And this depends on the specific context in which this strategy is applied. In certain cases, where the strategy is used to hide information that spontaneously emerges from the use of a system (for example communication patterns), the intent is to hide the information from anybody. In other cases, where information is collected, stored or processed legitimately by one party, the intent is to hide the information from any other, third, party. In this case, the strategy corresponds to ensuring confidentiality.

Common design patterns are the use of *encryption* (locally, or on the network using SSL), the use of *mix networks* to hide traffic patterns [6], or techniques to unlink

certain related events (e.g., anonymous cash [7] or *attribute based credentials* [4]). In essence, the HIDE strategy aims to achieve unlinkability and even unobservability [25].

4.3 Strategy #3: SEPARATE

The third design strategy is data or process separation. The strategy states that

The processing of personal information should be done in a distributed fashion whenever possible.

By separating the processing or storage of several sources of personal information that belong to the same person, complete profiles of one person cannot be made. The strategy of separation calls for distributed processing instead of centralised solutions. In particular, data from separate sources should be stored in separate databases, and these databases should not be linked if not needed. Data should be processed locally whenever possible, and stored locally if feasible as well. Database tables should be split when possible (and links between rows should be hard to find).

These days, with an emphasis on centralised, web based, services this strategy is often disregarded. However, the privacy guarantees offered by a decentralised social network like Diaspora¹⁴ are much more favourable than those of centralised approaches like Facebook and Google+. Further investigations into design pattern that implement the SEPARATE strategy are required, especially those that will satisfy business needs that usually steer towards a centralised solution.

4.4 Strategy #4: AGGREGATE

The forth design pattern, AGGREGATE, states that

Personal information should be processed at the highest level of aggregation and with the least possible detail in which it is (still) useful.

By restricting the amount of detail of personal information, or by considering this information at the group level instead of considering this information for each person separately, this personal information becomes less sensitive. When the information is sufficiently coarse grained, and the size of the group over which it is aggregated is sufficiently large, little information can be attributed to a single person, thus protecting its privacy.

Implicit design patterns are *aggregation over time* (for example used to provide some level of privacy protection

¹³ For brevity, we write personal information for personal identifiable information (PII), and we use term information processing to include the collection, storage and dissemination of that information as well.

¹⁴ <http://diasporafoundation.org/>

in smart metering and smart grid systems) or *dynamic location granularity* (used in location based services where the accuracy of the reported location of a user is adapted dynamically to ensure that a reasonable number of other users are at the same location). Sweeney's *k-anonymity* concept [28] is also a design pattern in this class.

4.5 Strategy #5: INFORM

The INFORM strategy corresponds to the important notion of transparency:

Data subjects should be adequately informed whenever personal information is processed.

Often, data protection regulation requires that data subjects are properly informed about the fact that personal information is processed when they use¹⁵ a certain system. The INFORM strategy underlines this fact. Data subjects should be informed about which information is processed, for what purpose, and by which means. This also includes information about the ways the information is protected, i.e. being open about the security of the system (the Kerckhoffs Principle). Data subjects should also be informed about third parties with which information is shared.

Possible design patterns in this category are the (nowadays pretty much defunct) *Platform for Privacy Preferences (P3P)*¹⁶ — although the latter could also fit the control strategy. *Data breach notifications* are also a design pattern in this category. Finally, Graf *et al.* [13] provide an interesting collection of privacy design patterns for informing the user from the Human Computer Interfacing perspective.

4.6 Strategy #6: CONTROL

The control strategy states that

Data subjects should have agency over the processing of their personal information.

The CONTROL strategy is in fact an important counterpart to the INFORM strategy. Without reasonable means of controlling the use of one's personal information, there is little use in informing a data subject about the fact that personal information is collected. Data protection legislation often gives the data subject the right to view, update and even ask the deletion of personal data collected

¹⁵ Or, less explicitly, *engage* with a system. A good example is entering an area with camera surveillance. This is not an explicit action to use that particular system. This broader understanding of 'using' a system is also important in ambient intelligent systems and the Internet of Things [17].

¹⁶ <http://www.w3.org/P3P/>

about him. This strategy underlines this fact, and design patterns in this class will give users the tools to exert their data protection rights.

Control goes beyond the strict implementation of data protection rights, however. It also governs the means by which users can decide whether to use a certain system, and the way they control what kind of information is processed about them. In the context of social networks, for example, the ease with which the user can update his privacy settings through the user interface determines the level of control to a large extent. So user interaction design is an important factor as well.

We are not aware of specific design patterns that fit this strategy, although methods to acquire informed consent, and certain user interaction design patterns could fit the bill.

4.7 Strategy #7: ENFORCE

The seventh strategy, ENFORCE, states:

A privacy policy compatible with legal requirements should be in place and should be enforced.

The ENFORCE strategy ensures that the system is compatible with data protection legislation, both at the time when the system is developed, as well as when the system is in operation. In this sense, enforcing purpose limitation is covered by this strategy as well. By specifying a privacy policy, and setting up the appropriate governance structures to enforce that policy, proper embedding of the IT system within the organisation is established.

Design patterns that implement this strategy could be certain types of *access control* (e.g., RBAC), and systems that implement privacy rights management (a form of digital rights management involving licenses to personal data, but then applied to privacy).

4.8 Strategy #8: DEMONSTRATE

The final strategy, DEMONSTRATE, requires a data controller to

Be able to demonstrate compliance with the privacy policy and any applicable legal requirements.

This strategy goes one step further than the ENFORCE strategy in that it requires the data controller to prove that it is in control. In particular this requires the data controller to be able to show how the privacy policy is effectively implemented within the IT system. In case of complaints or problems, he should immediately be able to determine the extent of any possible privacy breaches, for example.

Design patterns that implement this strategy are, for example, *privacy management systems*, and the use of logging and auditing.

5 Validation of our approach

To validate our approach, we verify that the eight privacy design strategies we derived cover the privacy principles of the ISO 29100 Privacy framework [19]. Moreover, we verify that the strategies can also be easily understood in the context of information flow systems.

5.1 The ISO 29100 Privacy Framework perspective

The ISO 29100 Privacy framework [19] suggests the following eleven privacy principles.

- *Consent and choice*: inform data subjects, present the available choices and obtain consent.
- *Purpose legitimacy and specification*: ensure compliance with data protection legislation and inform data subjects.
- *Collection limitation*: limit data collection to what is needed for the purpose.
- *Data minimisation*: minimise the amount of personal data collected, minimise the number of actors that have access, offer as default non-privacy invasive options, and delete data once it has become no longer necessary.
- *Use, retention and disclosure limitation*: limit the use, retention and disclosure of personal data to what is needed for the purpose.
- *Accuracy and quality*: ensure the data is accurate, up-to-date, adequate and relevant, verify this, and periodically check this.
- *Openness, transparency and notice*: inform data subjects about the data controller policies, give proper notices that personal data is being processed, provide information on how to access and review personal data.
- *Individual participation and access*: give data subjects the real possibility to access and review their personal data.
- *Accountability*: document policies, procedures and practices, assign the duty to implement privacy policies to specified individuals in the organisation, provide suitable training, inform about privacy breaches, give access to effective sanctions and procedures for compensations in case of privacy breaches.
- *Information security*: provide a proper level of security, and implement the right controls, based on an appropriate risk assessment.
- *Privacy compliance* verify and demonstrate that the IT systems meets legal requirements, and have appropriate internal controls and supervision mechanisms.

We will not discuss the merits of this subdivision in any depth, although we do note that there is quite a bit of overlap between *Consent and choice*, *Purpose specification*

and *Openness, transparency and notice*. Similarly, *Collection limitation*, *Data minimisation* and *Use, retention and disclosure limitation* all more or less describe the same need for data minimisation. And in our opinion, even though data accuracy and quality are vastly important (especially in health care), we do feel that data accuracy and quality are not proper aspects of privacy protection. In fact, poor data quality may protect privacy, and in fact spreading disinformation has been proposed as a protective measure [31] (for example to prevent profiling)¹⁷.

The ISO 29100 principles lie somewhere between purely legal requirements, and the more technically oriented design strategies that we aim to develop here. But we can map most of the privacy principles to the design strategies derived so far. For example *consent and choice* can be implemented using the `INFORM` strategy (except for obtaining choice, which is covered by the `CONTROL` strategy). *Purpose specification* can be achieved using the `INFORM` strategy. *Collection limitation*, *Data minimisation* and *Use, retention and disclosure limitation* all are achieved using either the `MINIMISE`, `SEPARATE` or `AGGREGATE` strategy. *Openness, transparency and notice* is achieved using the `INFORM` strategy and *Individual participation and access* using the `CONTROL` strategy. *Accountability* like *Information security* corresponds to the `ENFORCE` strategy (possibly supported by the `HIDE` strategy). Finally, *Privacy compliance* corresponds to the `DEMONSTRATE` strategy.

This leaves *data quality* (for which we already argued that this does not contribute to privacy protection), and *purpose legitimacy*. According to us, the latter is not a privacy design strategy by itself, but rather an overarching legal principle to which each of the privacy design strategies contribute.

5.2 The OECD privacy principles

The Organisation of Economic Co-Operation and Development (OECD) published a set of privacy guidelines in 1980 [22], of which the US fair information practices (FIPs) [29] — *notice, choice, access and security* — are a subset. In fact, the OECD defined the following principles: Collection Limitation, Data Quality, Purpose Specification, Use Limitation, Security Safeguards, Openness, Individual Participation, and Accountability. These principles overlap to a very large extent the ISO 29001 privacy principles, which implies that also the OECD privacy principles can be mapped to our privacy design strategies.

¹⁷ Another example is the TrackMeNot browser plugin <http://cs.nyu.edu/trackmenot/>, described in [18].

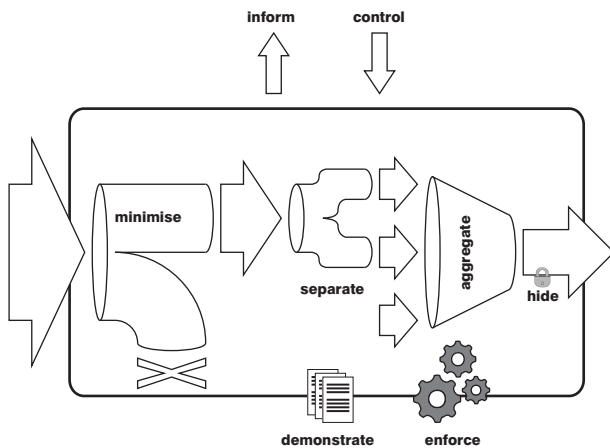


Fig. 2 The process flow metaphor of the eight privacy design strategies.

5.3 Information flow systems

These eight strategies also naturally apply to an information flow system, where data is not stored but flows instead, as illustrated in Figure 2. In this view, MINIMISE corresponds to processing only a selected subset of the incoming data (and throwing the rest away), while SEPARATE corresponds to splitting the data stream in several parts that are each further processed at separate locations. AGGREGATE corresponds to combining (and compressing) data streams, while HIDE (for example) encrypts the data while in transit. INFORM, CONTROL, ENFORCE and DEMONSTRATE are essentially the same as in the information storage model.

6 Conclusions

We have defined the concept of a design strategy, and derived eight privacy design strategies based on the legal perspective on privacy. We have described these strategies in some detail, and have provided a first insight into the possible privacy design patterns that contribute to these privacy design strategies. Finally, we have validated our approach by verifying that the classification covers the privacy principles postulated by ISO 29100 [19] and the OECD [22], and that the classification applies to both information storage and information flow systems.

Missing from the set of privacy design strategies is a VERIFY strategy that ensures data accuracy and data quality. Although data quality is listed as a separate principle in both the ISO 29001 framework [19] and the OECD guidelines [22], we believe this is not a privacy related issue. In fact, spreading disinformation and confusing ob-

servers is a method that is sometimes used to protect privacy.

We have taken the legal perspective as point of departure in our approach, and have validated our results against both the technological and the privacy policy perspective. We have not taken into account any philosophical, sociological or values-based perspectives. It would be interesting to investigate whether these perspectives have any impact on the list of privacy design strategies reported here.

This paper discusses work in progress. In particular, further research will be performed to classify existing privacy design patterns into privacy design strategies, and to describe these design patterns in more detail (using a uniform template). Moreover, we have identified several implicitly defined design patterns (like *attribute based credentials*) that arise from our study of existing privacy enhancing technologies. Finally, our classification could also contribute to improving the classification of privacy principles given in ISO 29100 [19], especially in limiting the amount of overlap among the several principles.

Further developments and collaboration in this line of research will also be documented on our Wiki <http://wiki.science.ru.nl/privacy/>. We would very much welcome contributions from others here.

References

1. J. Borking. Der identity-protector. *Datenschutz und Datensicherheit*, 20(11):654–658, 1996.
2. Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy*. MIT Press, 1st edition, 2000. ISBN 0-262-02491-8.
3. Frank Buschmann, Regine Meunier, Hans Rohnert, and Peter Sommerlad. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, 1996.
4. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfizmann, editor, *EUROCRYPT*, LNCS 2045, pages 93–118, Innsbruck, Austria, May 6–10 2001. Springer.
5. Ann Cavoukian. Privacy by design – the 7 foundational principles. Technical report, Information and Privacy Commissioner of Ontario, jan 2011. (revised version).
6. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.
7. David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO*, LNCS 403, pages 319–327, Santa Barbara, California, August 21–25 1988. Springer.
8. Communication COM (2007)228 from the Commission to the European Parliament and the Council. On Promoting Data Protection by Privacy Enhancing Technologies (PETs). (*Not published in the OJC*), 2007.
9. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995. On the protection of individuals with regard to the processing of personal data and on the free movement of such data. *OJ C L*, 281:0031 – 0050, November 23 1995.

10. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
11. Ian Goldberg. Privacy-enhancing technologies for the internet, ii: Five years later. In Roger Dingledine and Paul F. Syverson, editors, *Privacy Enhancing Technologies*, LNCS 2482, pages 1–12, San Francisco, CA, USA, April 14–15 2002. Springer.
12. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
13. Cornelia Graf, Peter Wolkstorfer, Arjan Geven, and Manfred Tscheligi. A pattern collection for privacy enhancing technology. In *The 2nd Int. Conf. on Pervasive Patterns and Applications (PATTERNS 2010)*, Lisbon, Portugal, November 21–26 2010.
14. Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design. In *Conference on Computers, Privacy & Data Protection (CPDP 2011)*, 2011.
15. Munawar Hafiz. A collection of privacy design patterns. In *Proceedings of the 2006 conference on Pattern languages of programs, PLoP '06*, pages 7:1–7:13, New York, NY, USA, 2006. ACM.
16. Munawar Hafiz. A pattern language for developing privacy enhancing technologies. *Softw. Pract. Exper.*, 2011. doi: 10.1002/spe.1131.
17. Jaap-Henk Hoepman. In things we trust? towards trustability in the internet of things, September 2011. eprint CoRR cs.CR:1109.2637.
18. D. Howe and H. Nissenbaum. Trackmenot: Resisting surveillance in web search. In I. Kerr, C. Lucock, and V. Steeves, editors, *Lessons From the Identity Trail: Privacy, Anonymity and Identity in a Networked Society*, chapter 23. Oxford University Press, 2009.
19. ISO/IEC 29100. Information technology – Security techniques – Privacy framework. Technical report, ISO JTC 1/SC 27.
20. B. Jacobs. Select before you collect. *Ars Aequi*, 54:1006–1009, December 2005.
21. P. Kruchten. An ontology of architectural design decisions. In Jan Bosch, editor, *Proc. of the 2nd Groningen Workshop on Software Variability Management*, Groningen, The Netherlands, 2004.
22. Organisation of Economic Co-Operation and Development. OECD guidelines on the protection of privacy and transborder flows of personal data, 1980.
23. Siani Pearson and Azzedine Benameur. Decision support for design for privacy: A system focused on privacy by policy. In *PrimeLife/IFIP Summer School 2010: Privacy and Identity Management for Life*, Helsingborg, Sweden, August 2010. (to appear).
24. Siani Pearson and Yun Shen. Context-aware privacy design pattern selection. In Sokratis K. Katsikas, Javier Lopez, and Miguel Soriano, editors, *Trust, Privacy and Security in Digital Business (TrustBus)*, 7th International Conference, LNCS 6264, pages 69–80, Bilbao, Spain, August 30–31 2010. Springer.
25. Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology (version v0.34 aug. 10, 2010). http://dud.inf.tu-dresden.de/Anon_Terminology.shtml.
26. D. J. Solove. A taxonomy of privacy. *University of Pennsylvania Law Review*, 154(3):477–564, 2006.
27. Sarah Spiekermann and Lorrie Faith Cranor. Engineering privacy. *IEEE Trans. Software Eng.*, 35(1):67–82, 2009.
28. Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
29. US Federal Trade Commission. Privacy online: Fair information practices in the electronic marketplace, a report to congress, 2000.
30. G. W. van Blarckom, J. J. Borking, and P. Verhaar. PET. In G. W. van Blarckom, J. J. Borking, and J. G. E. Olk, editors, *Handbook of Privacy and Privacy-Enhancing Technologies - The case of Intelligent Software Agents*, chapter 3, pages 33–54. College bescherming persoonsgegevens, The Hague, The Netherlands, 2003.
31. Steven Euijong Whang and Hector Garcia-Molina. Disinformation techniques for entity resolution. Technical report, Stanford University, 2011.